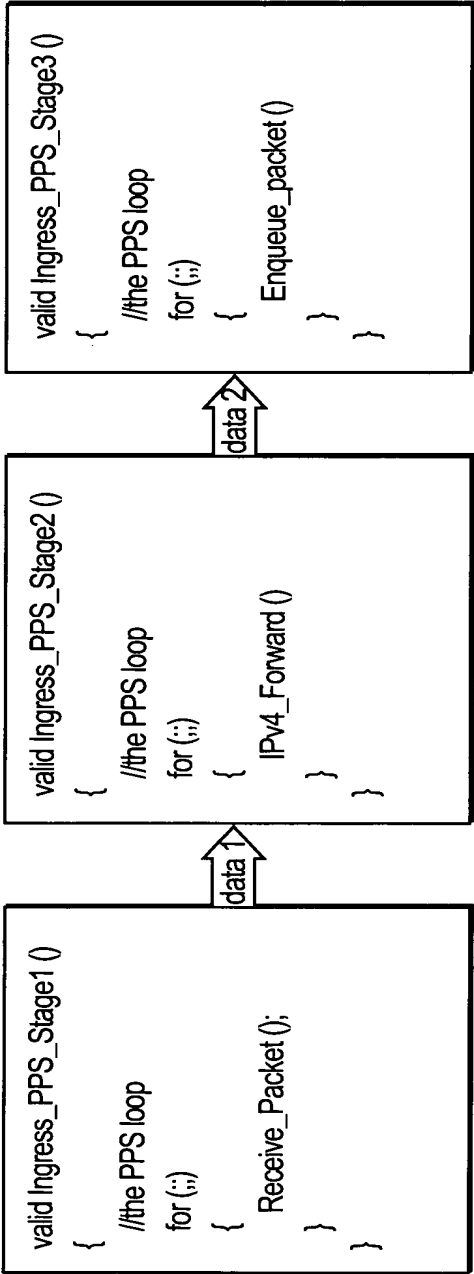


**FIG. 1**

```
valid Ingress_PPS ()
{
    //the PPS loop
    for (;;)
    {
        Receive_Packet ();
        IPv4_Forward ()
        Enqueue_packet ();
    }
}
```

FIG. 2A

SEQUENTIAL PPS 280



PIPELINED PPS 300

FIG. 2B

```

void MyPPS()
{
    for (;;)
    {
        if (p ())
        {
            x = f1 ();
            y = g1 ();
        }
        else
        {
            x = f2 ();
            y = g2 ();
        }
        z = h (x,y);
    }
}
    
```

290

FIG. 3A

```

void MyPPS2()
{
    for (;;)
    {
        if (p ())
        {
            x1 = f1 ();
            y1 = g1 ();
        }
        else
        {
            x2 = f2 ();
            y2 = g2 ();
        }
        x = phi (x1, x2);
        y = phi (y1, y2);
        z = h (x, y);
    }
}
    
```

400

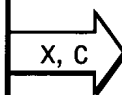
FIG. 4

```

void MyPPS2_Stage1 ()
{
    for (;;)
    {
        if (p ())
        {
            c = 1;
            x = f1 ();
        }
        else
        {
            c = 2;
            x = f2 ();
        }
        Send_To_Stage2 (c, x);
    }
}
    
```

310

FIG. 3B



```

void MyPPS2_Stage2 ()
{
    for (;;)
    {
        Receive_From_Stage1 (&c, &x);
        if (C ==1)
        {
            y = g1 ();
        }
        else
        {
            y = g2 ();
        }
        z = h (x,y);
    }
}
    
```

320

FIG. 3C

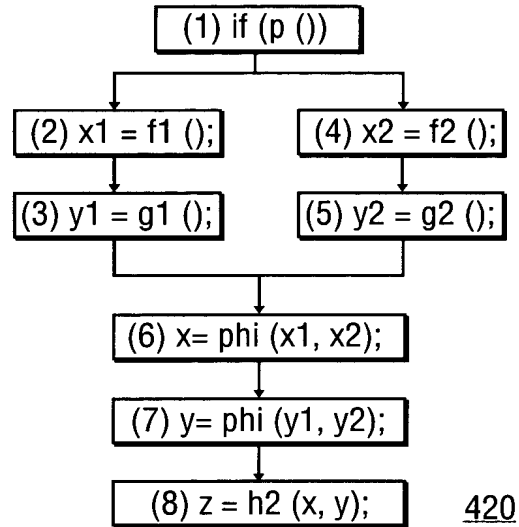


FIG. 5

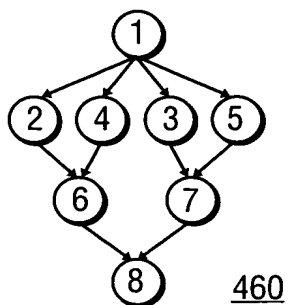


FIG. 6

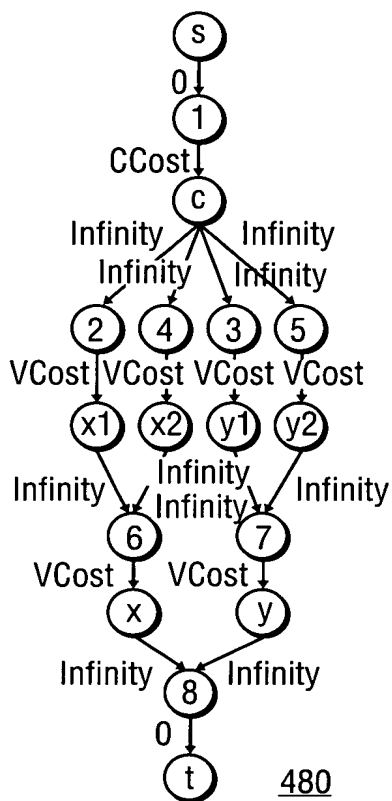
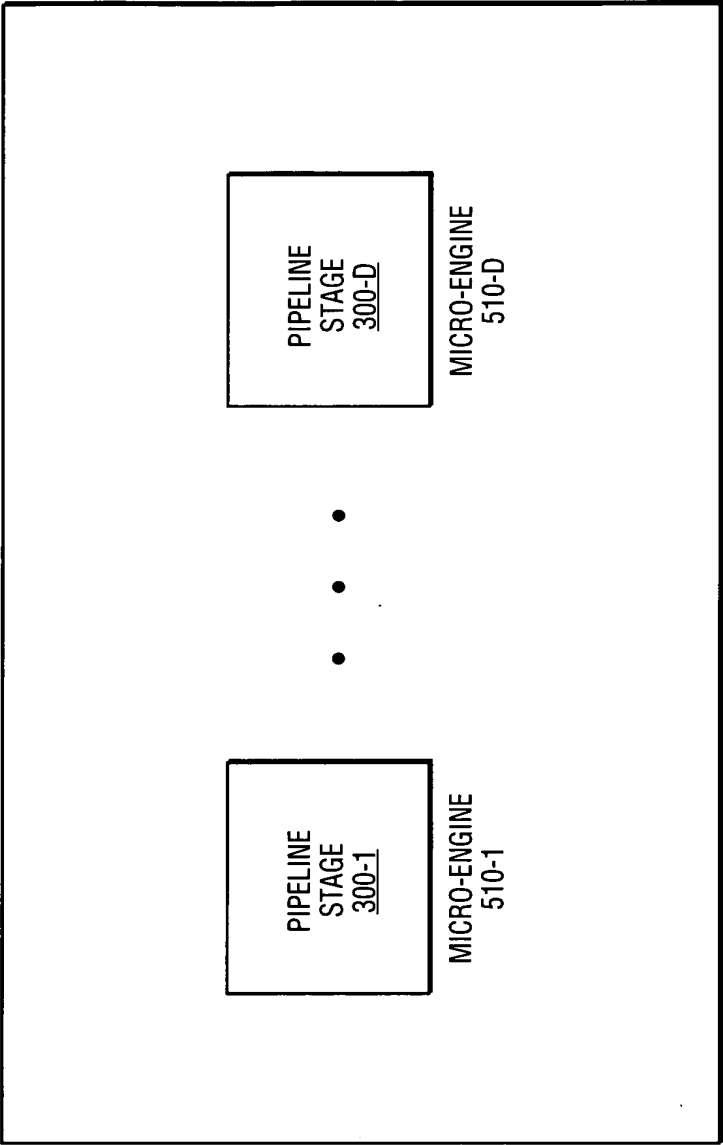


FIG. 7



NP  
500

FIG. 8

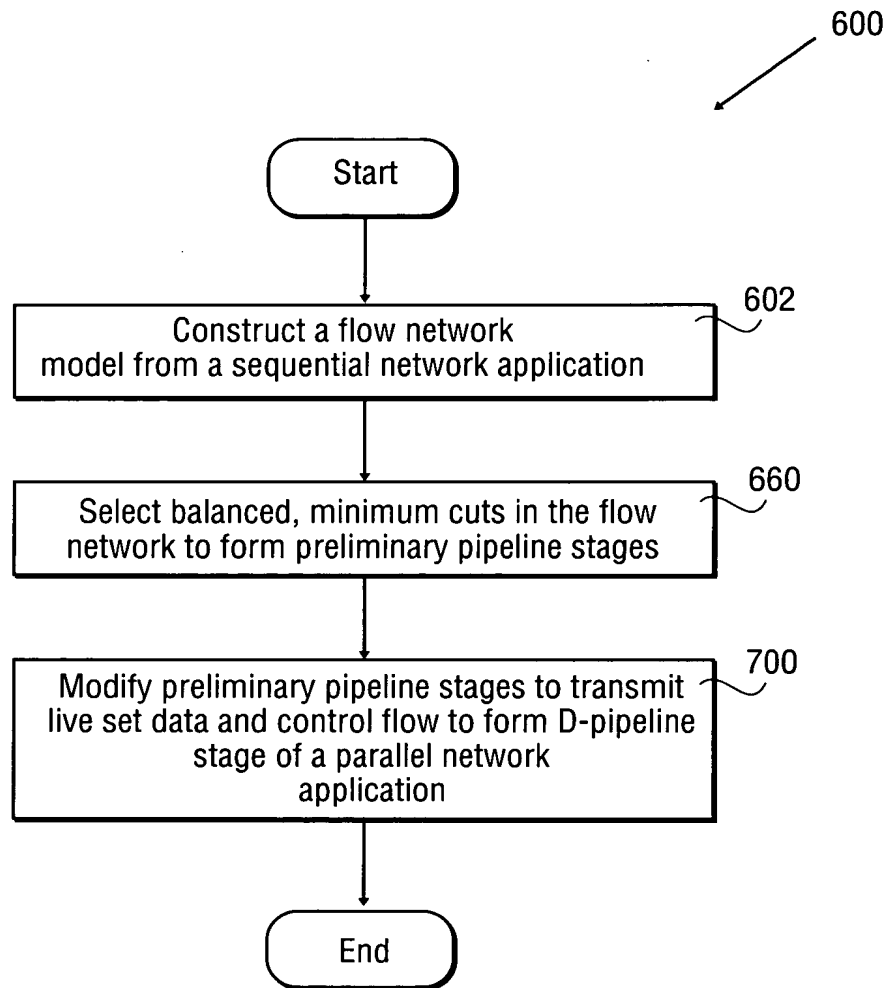


FIG. 9

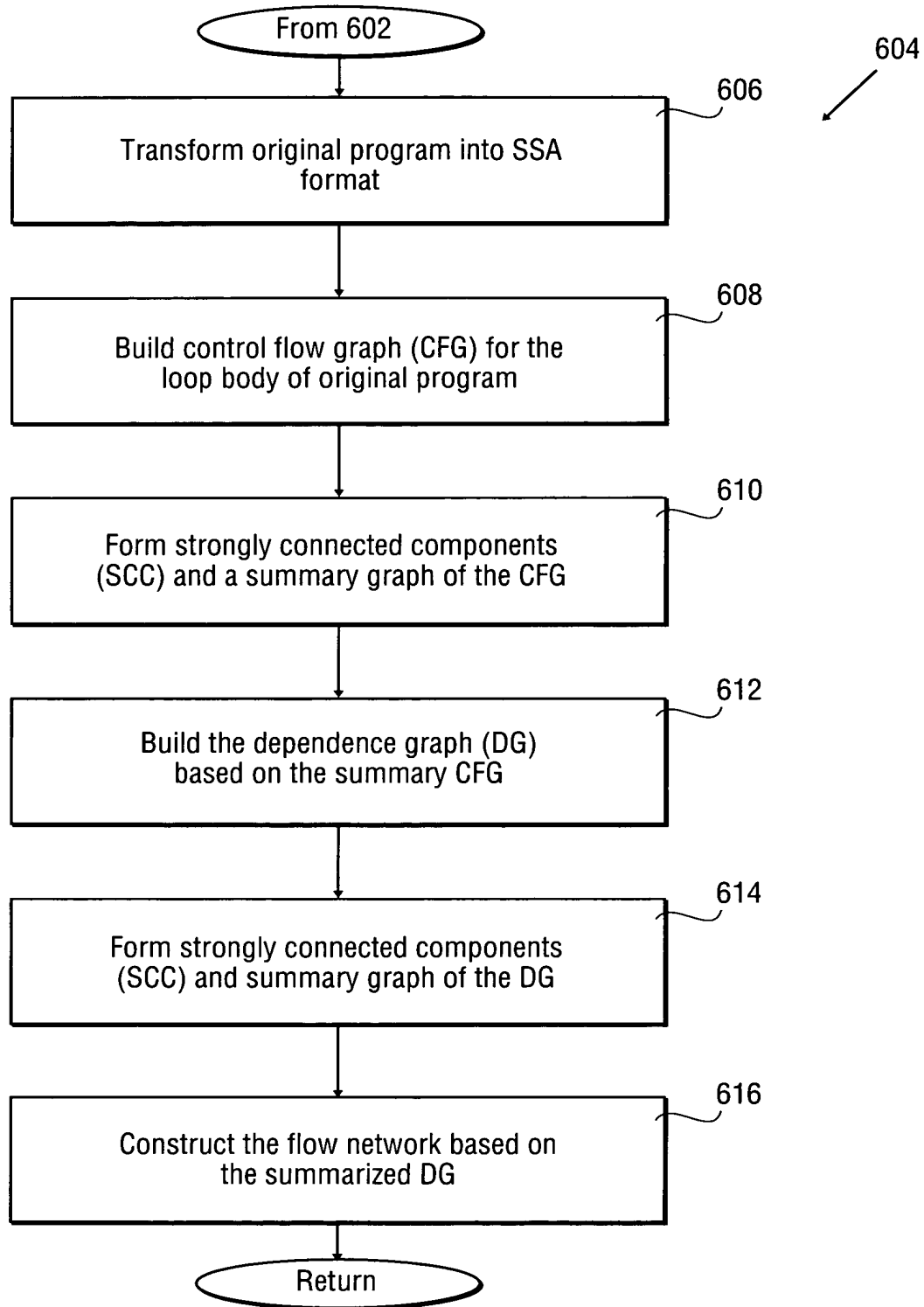


FIG. 10



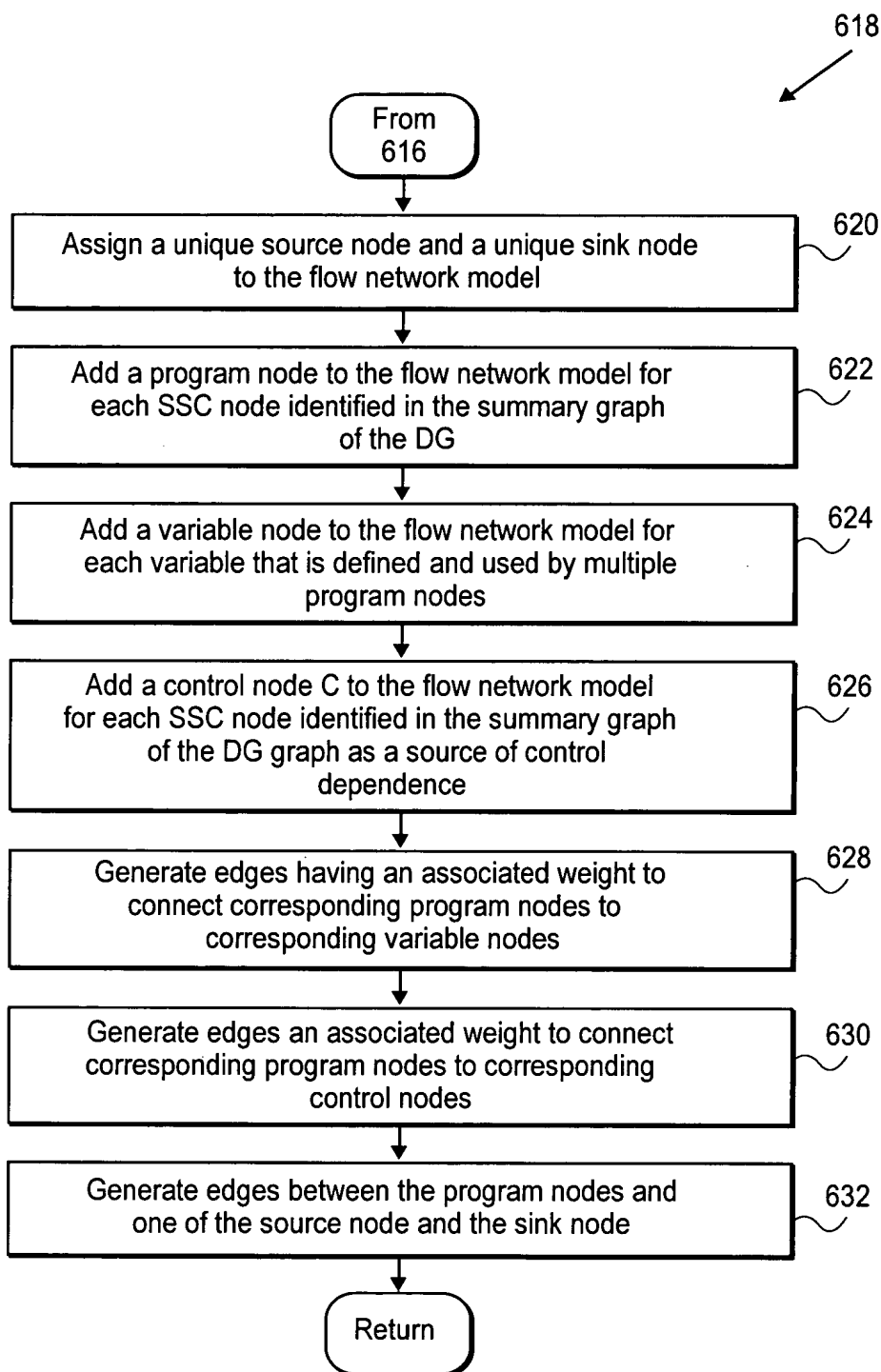


FIG. 11

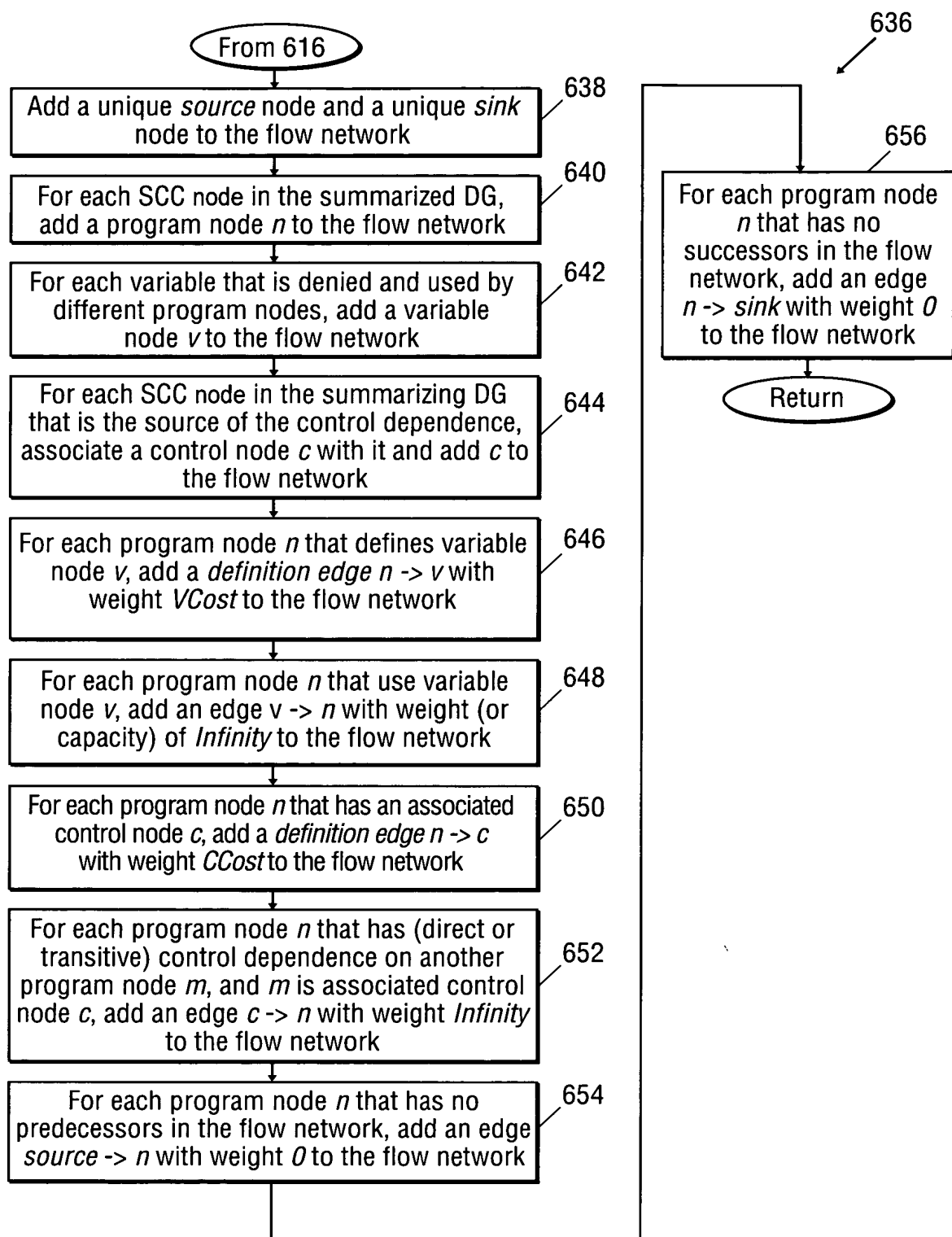


FIG. 12

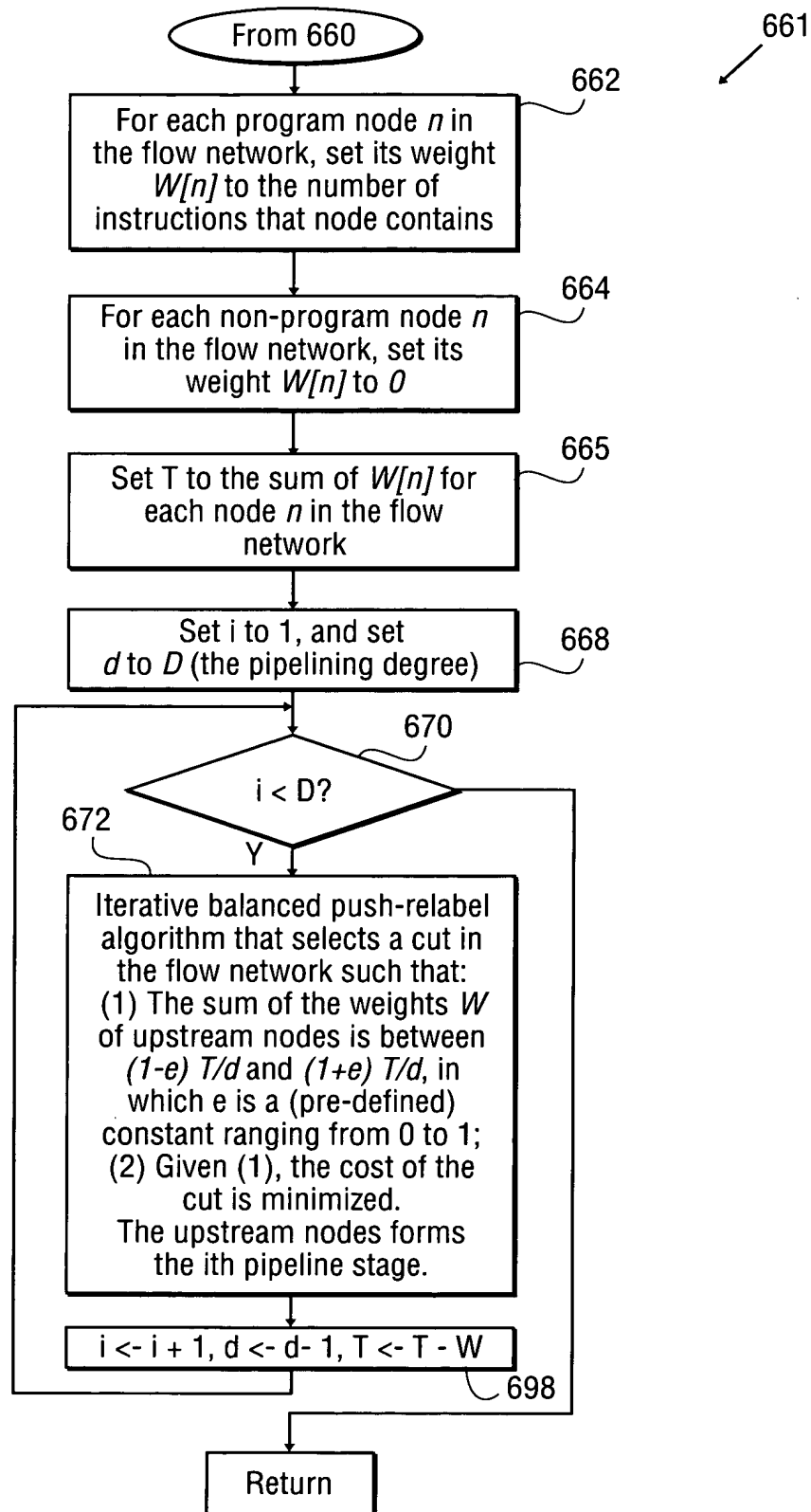
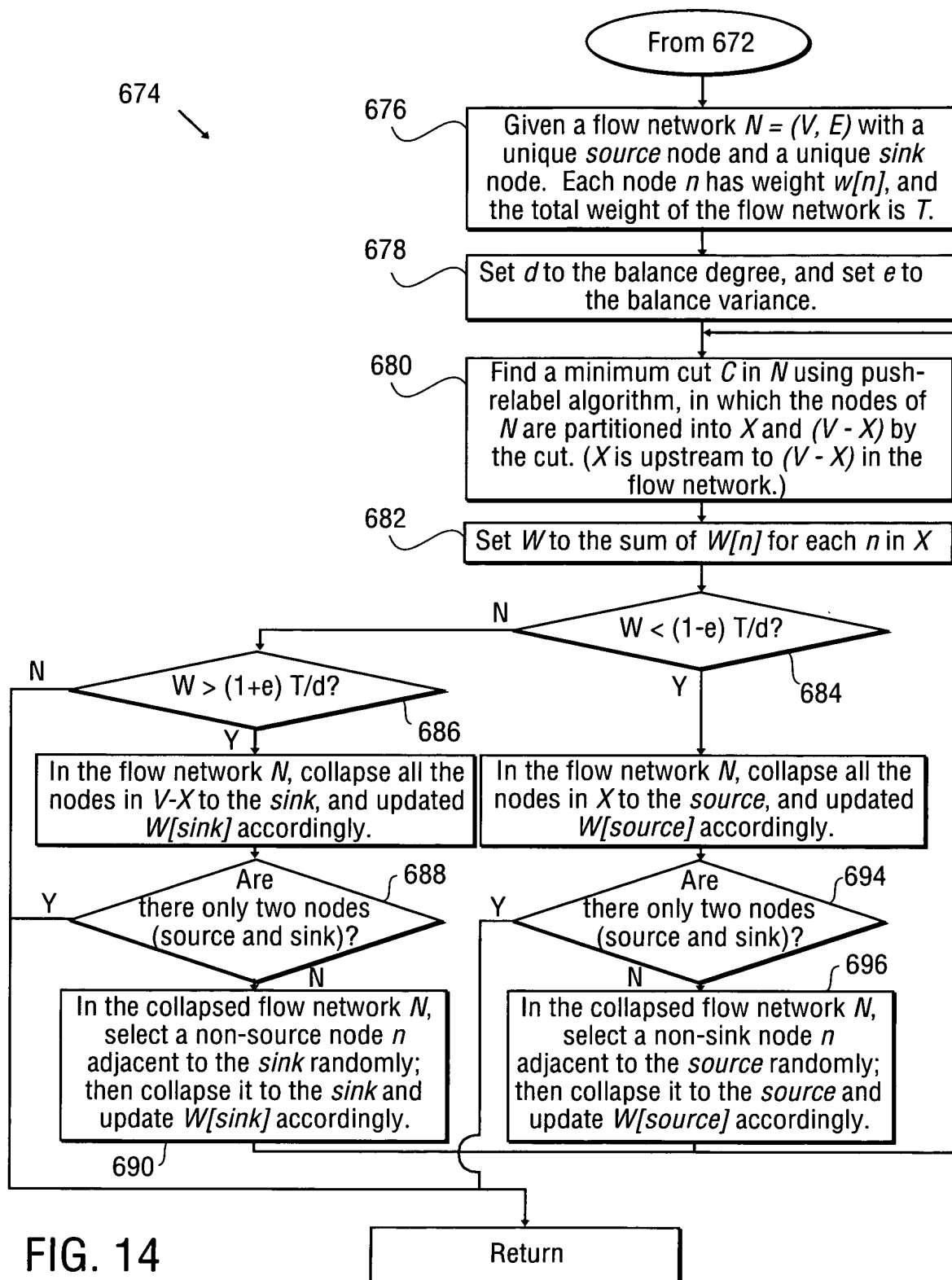


FIG. 13



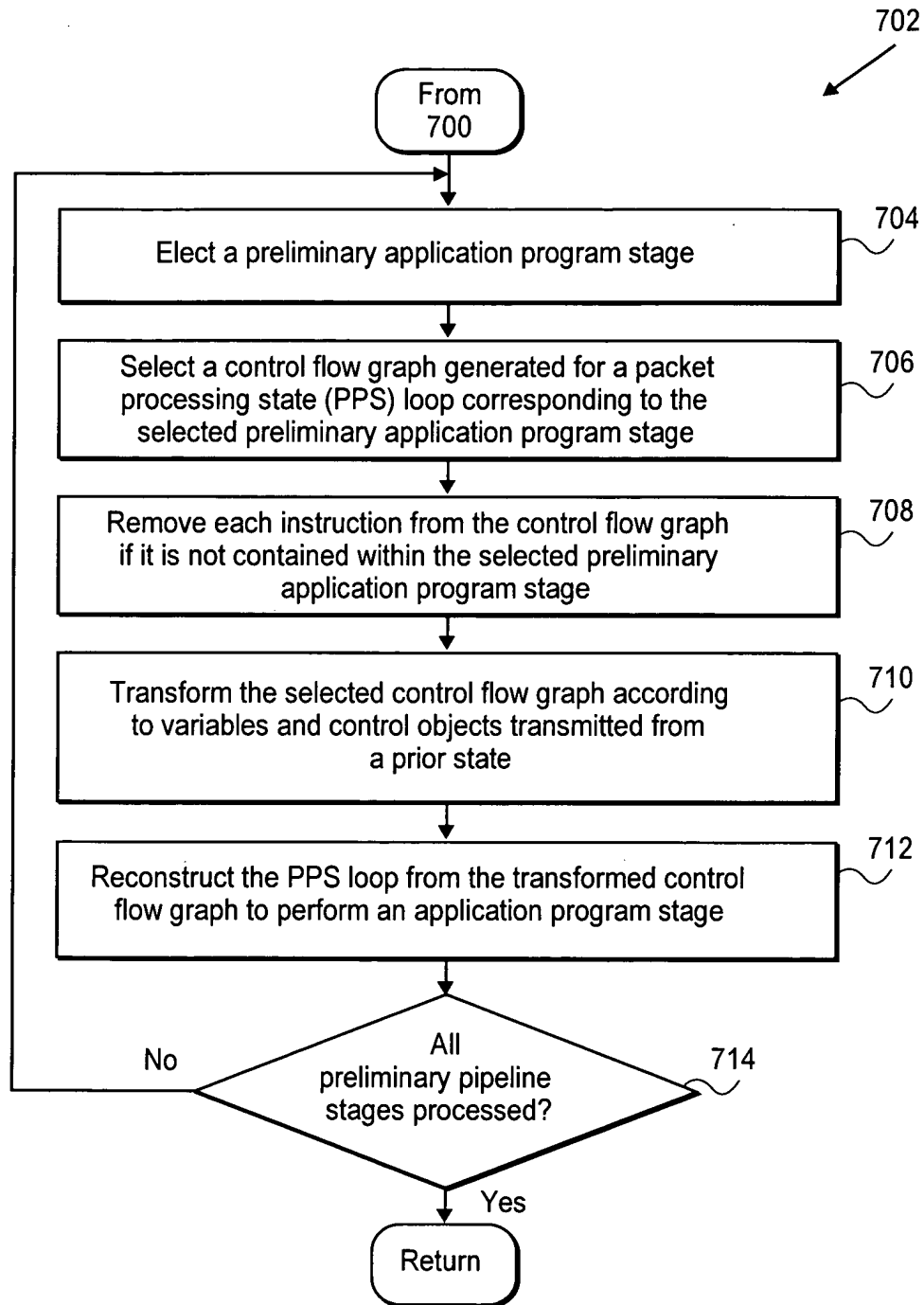


FIG. 15

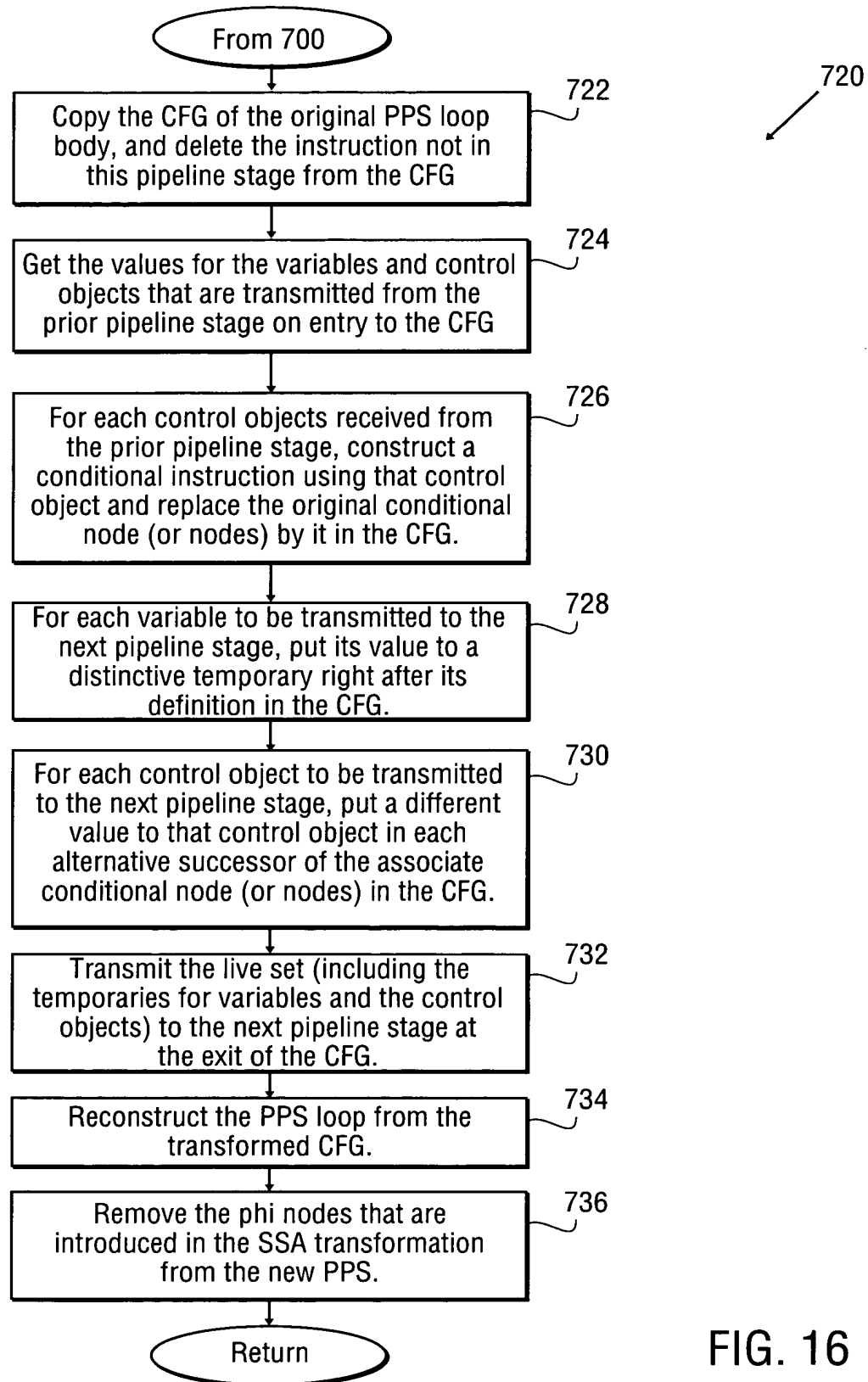


FIG. 16